FRAUD DETECTION IN PYTHON

Normal versus abnormal behaviour

Charlotte Werger
Data Scientist

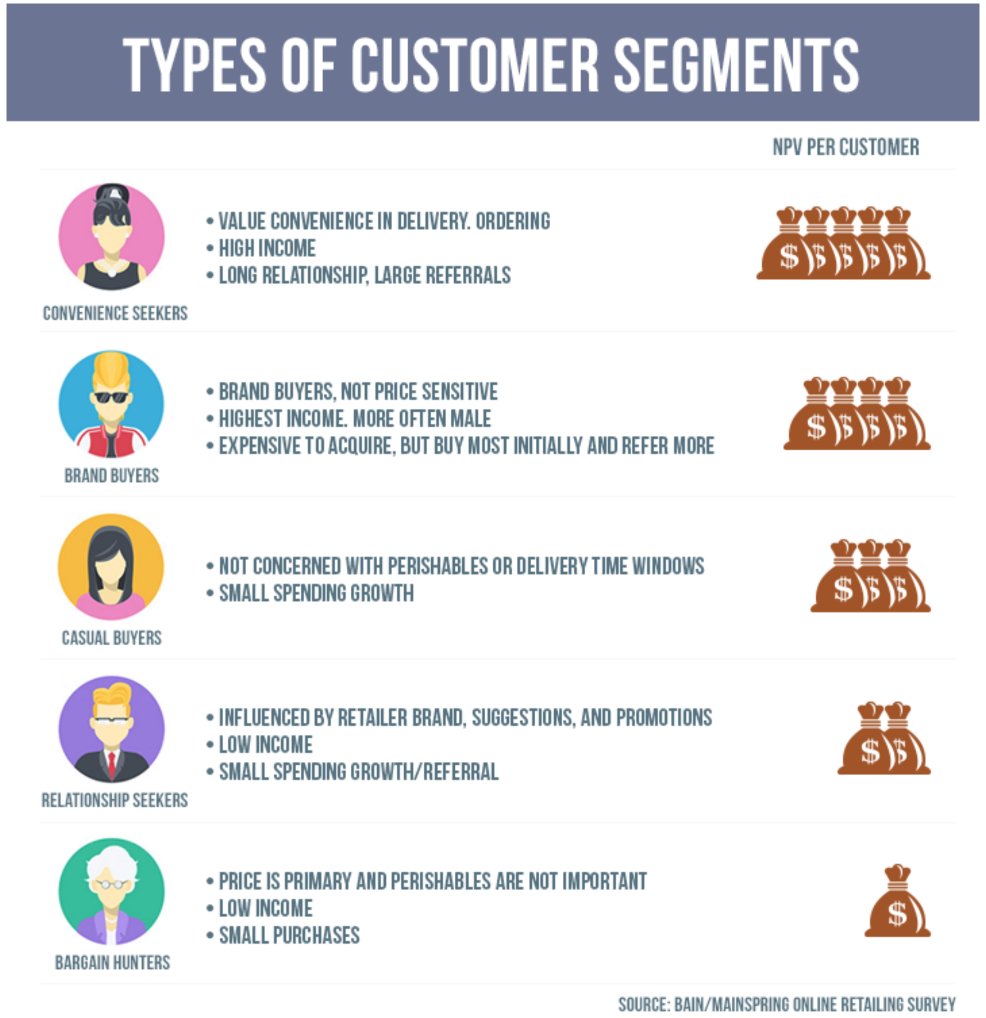
Fraud detection without labels

- Using unsupervised learning to distinguish normal from abnormal behaviour
- Abnormal behaviour by definition is not always fraudulent
- Challenging because difficult to validate
- But...realistic because very often you don't have reliable labels

What is normal behaviour?

- Thoroughly describe your data: plot histograms, check for outliers, investigate correlations and talk to the fraud analyst
- Are there any known historic cases of fraud? What typifies those cases?
- Normal behaviour of one type of client may not be normal for another
- Check patterns within subgroups of data: is your data homogenous?

Customer segmentation: normal behaviour within segments



FRAUD DETECTION IN PYTHON

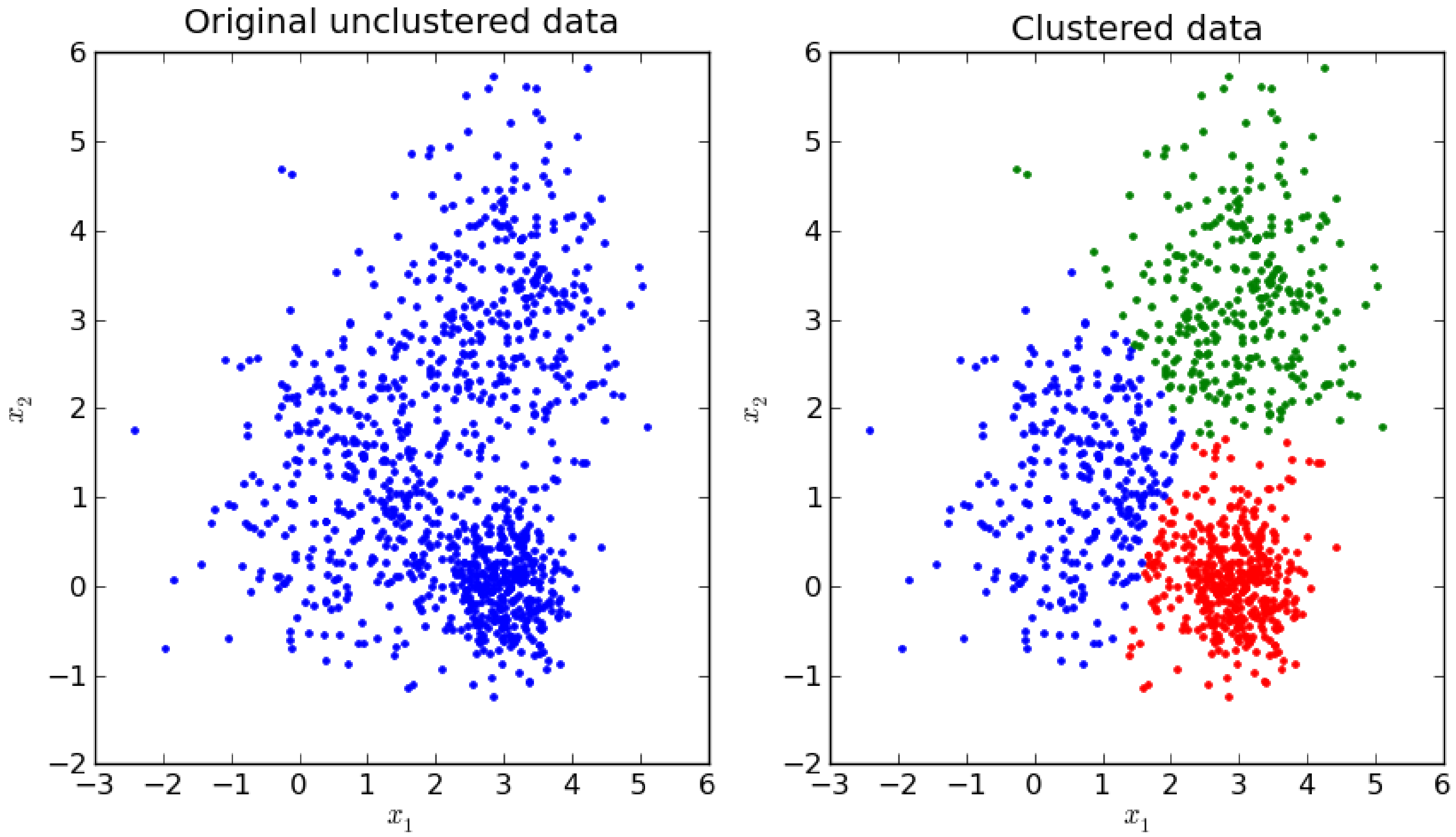
Let's practice!

FRAUD DETECTION IN PYTHON

Refresher on clustering methods

Charlotte Werger
Data Scientist

Clustering: trying to detect patterns in data

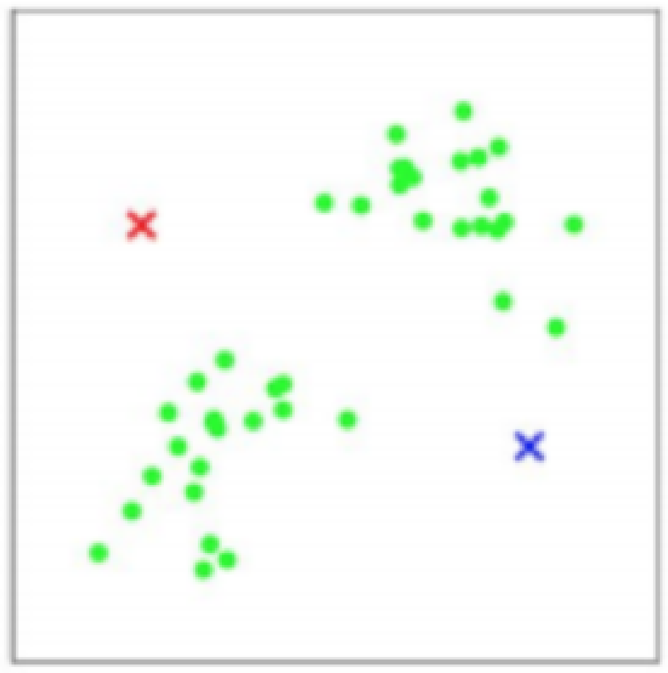


K-means clustering: using the distance to cluster centroids

A

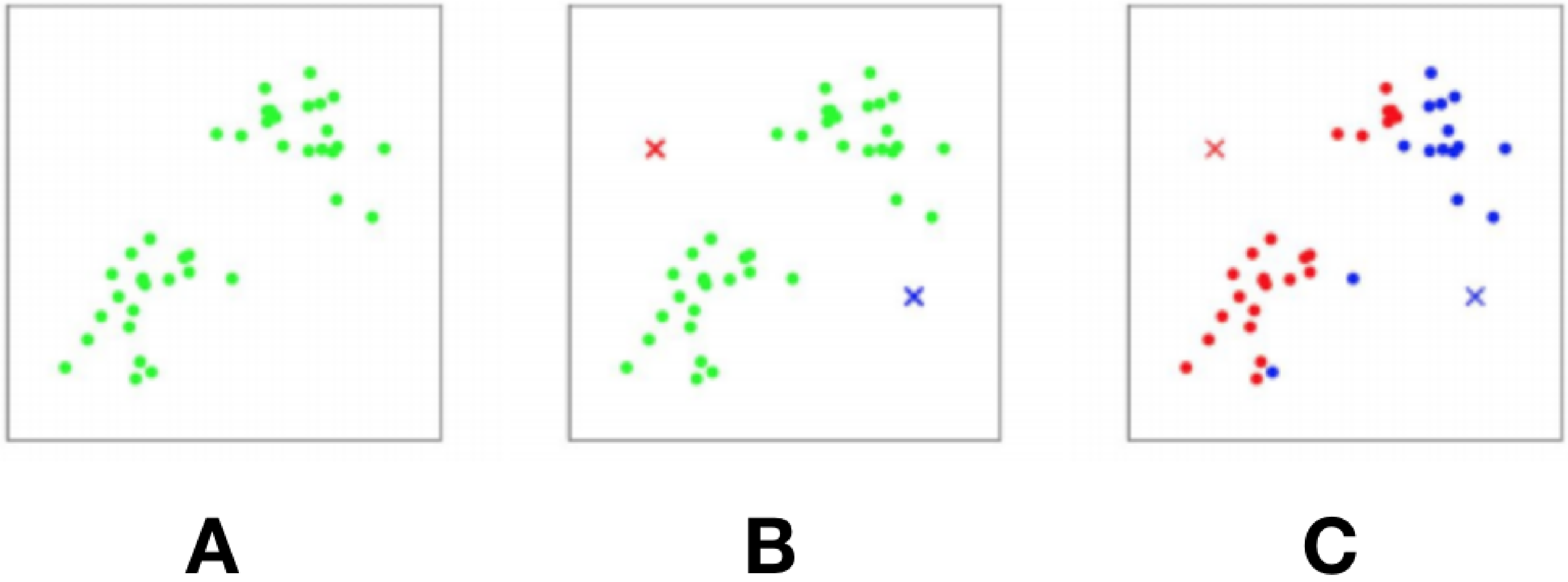
K-means clustering: using the distance to cluster centroids

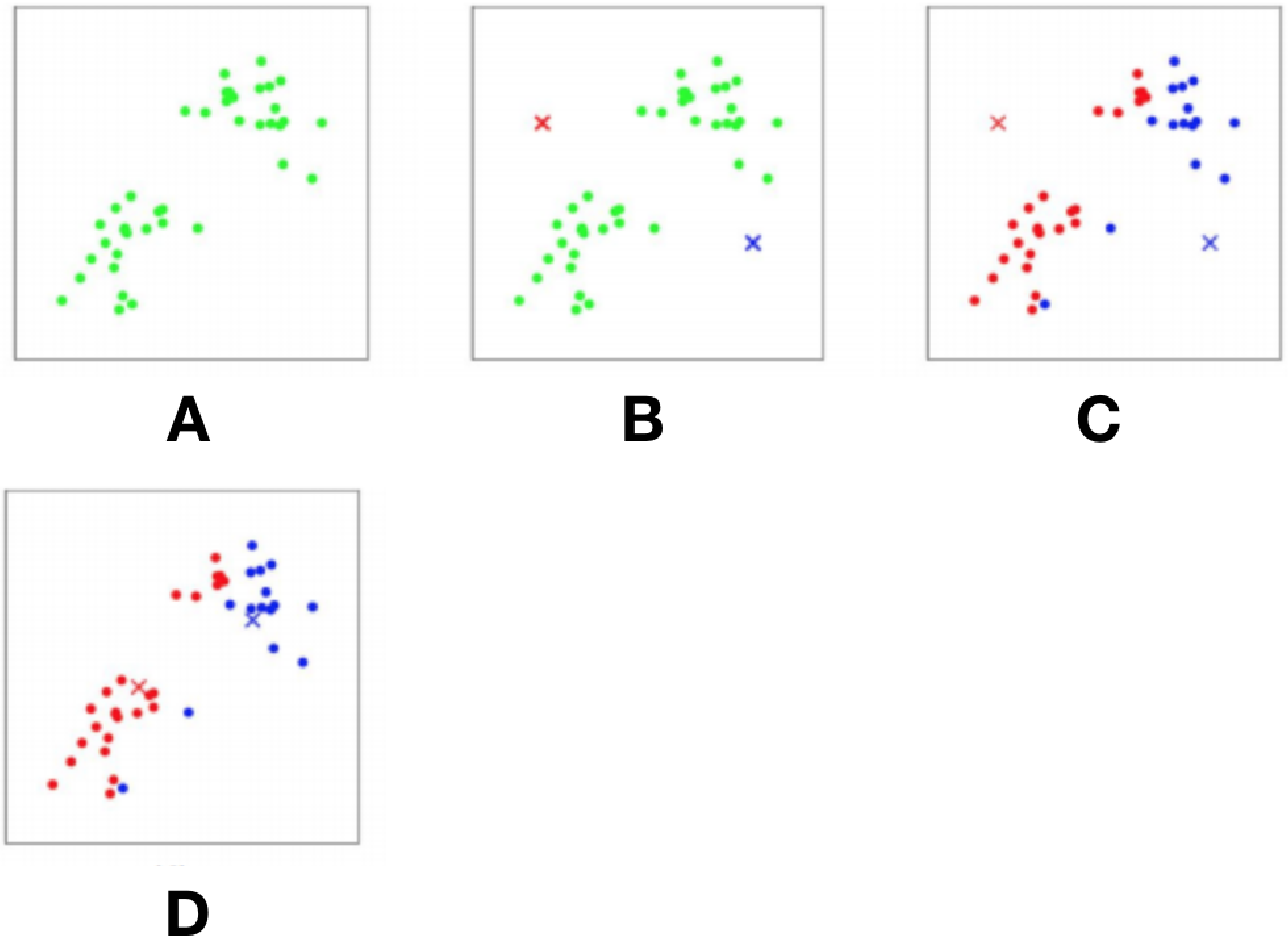
A

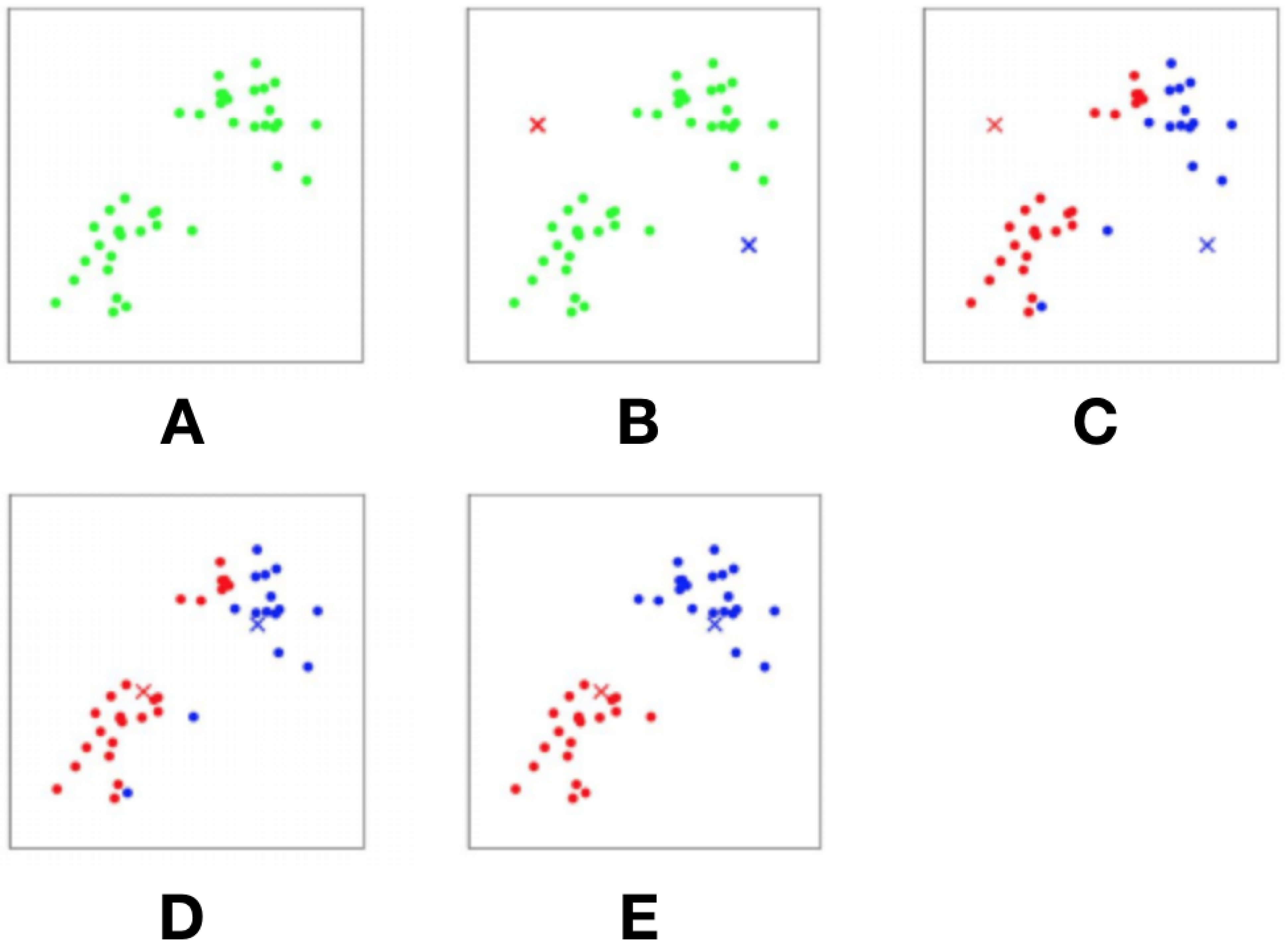


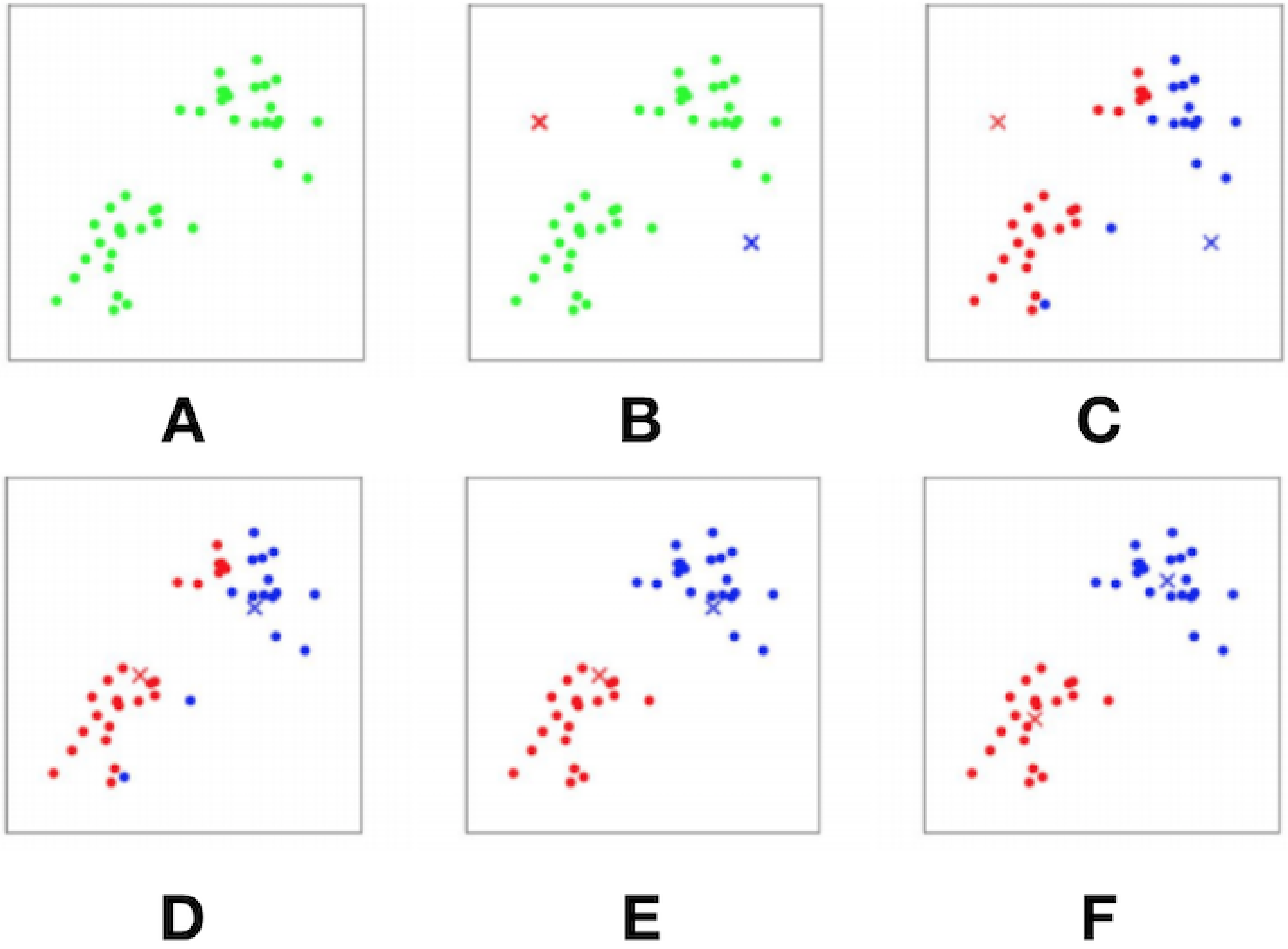
B

K-means clustering: using the distance to cluster centroids




A
B
C
D


A
B
C
D
E


A
B
C
D
E
F

K-means clustering in Python

```
# Import the packages
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans

# Transform and scale your data
X = np.array(df).astype(np.float)

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Define the k-means model and fit to the data
kmeans = KMeans(n_clusters=6, random_state=42).fit(X_scaled)
```

The right amount of clusters

Checking the number of clusters:

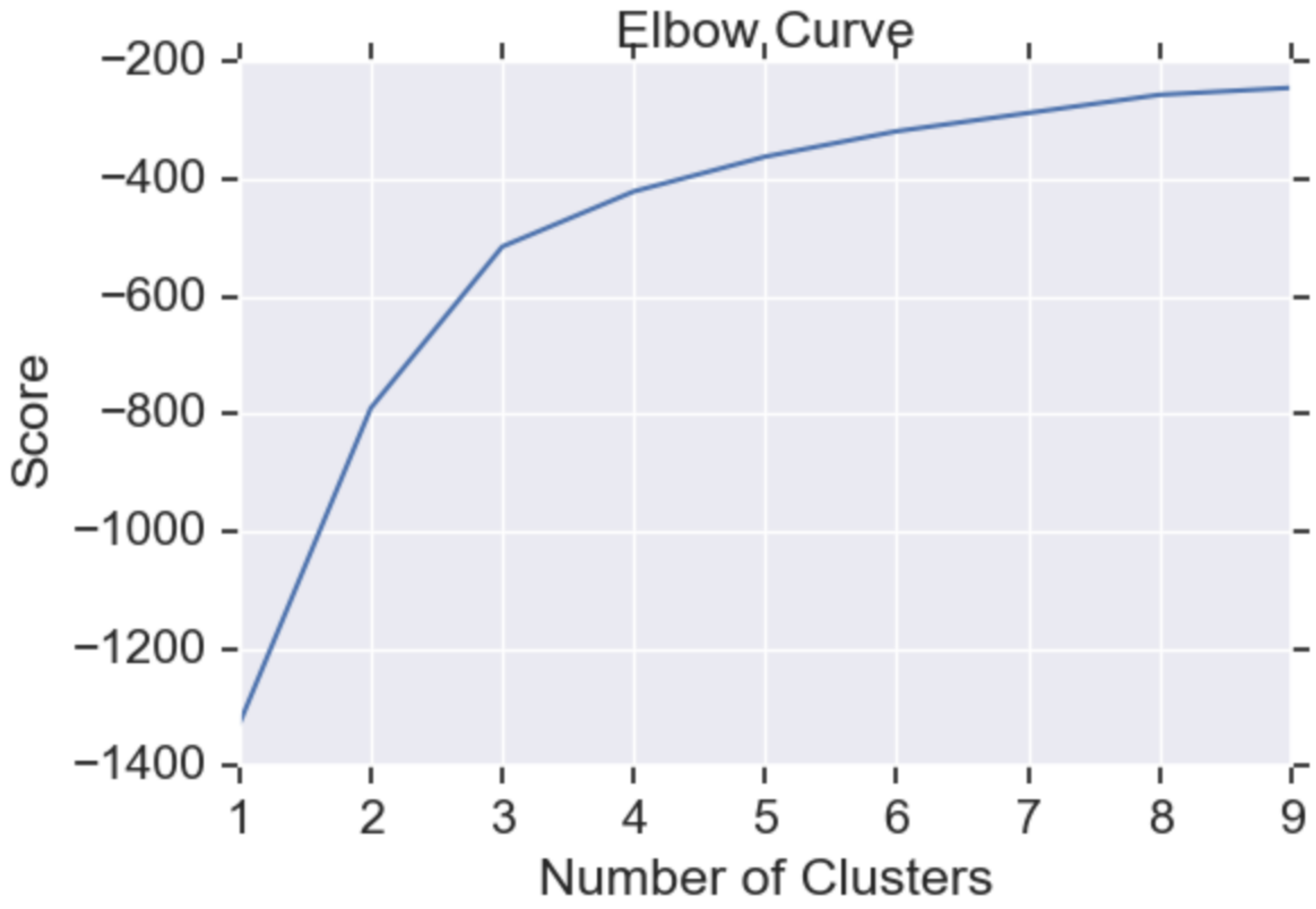
- Silhouette method
- Elbow curve

```
clust = range(1, 10)
kmeans = [KMeans(n_clusters=i) for i in clust]

score = [kmeans[i].fit(X_scaled).score(X_scaled) for i in range(len(kmeans)
```

```
plt.plot(clust,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

The Elbow Curve



FRAUD DETECTION IN PYTHON

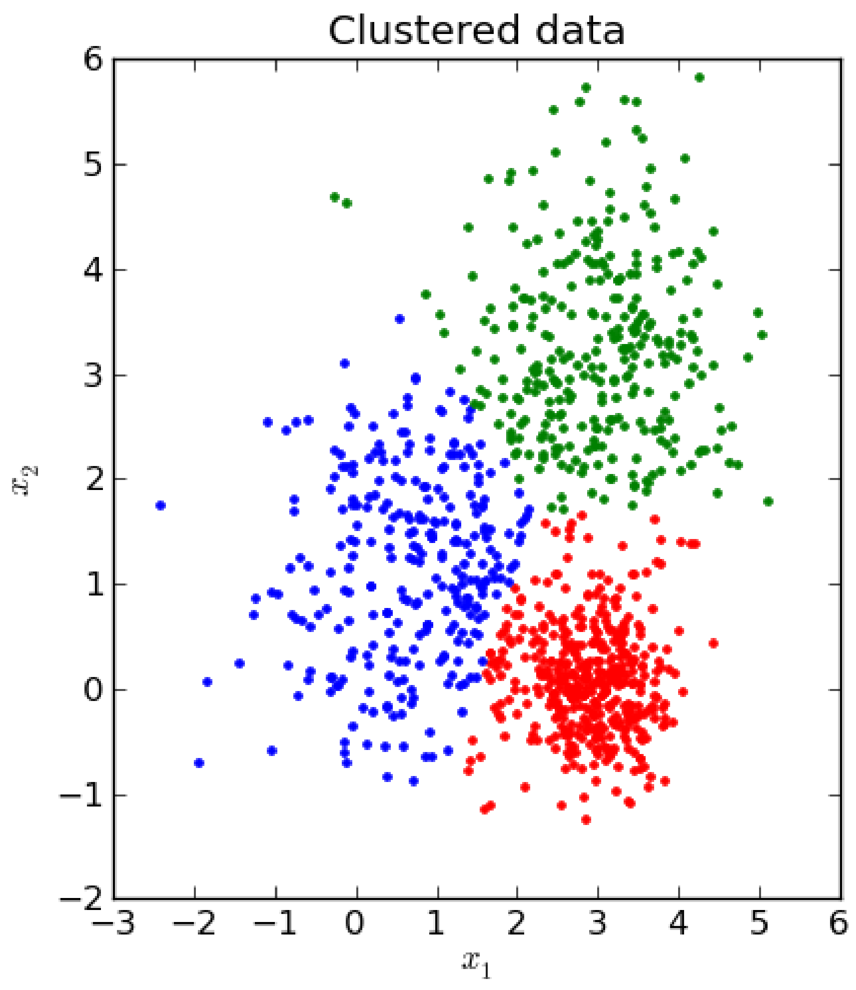
Let's practice!

FRAUD DETECTION IN PYTHON

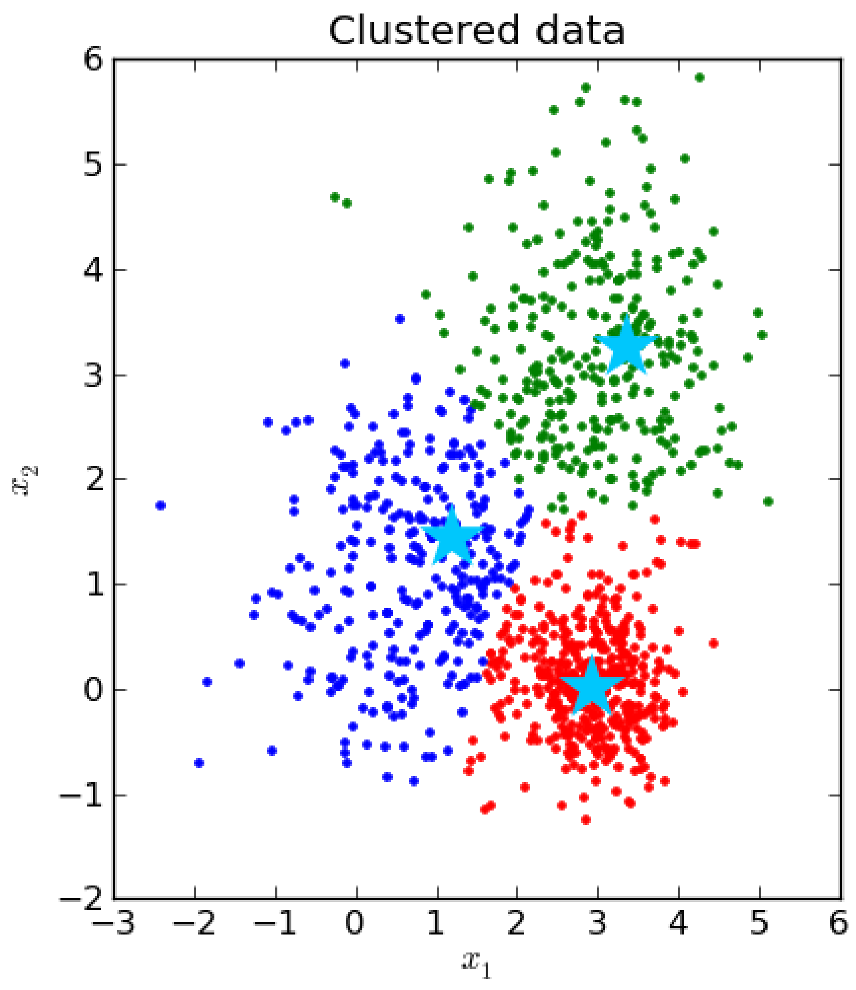
Assigning fraud versus non-fraud cases

Charlotte Werger
Data Scientist

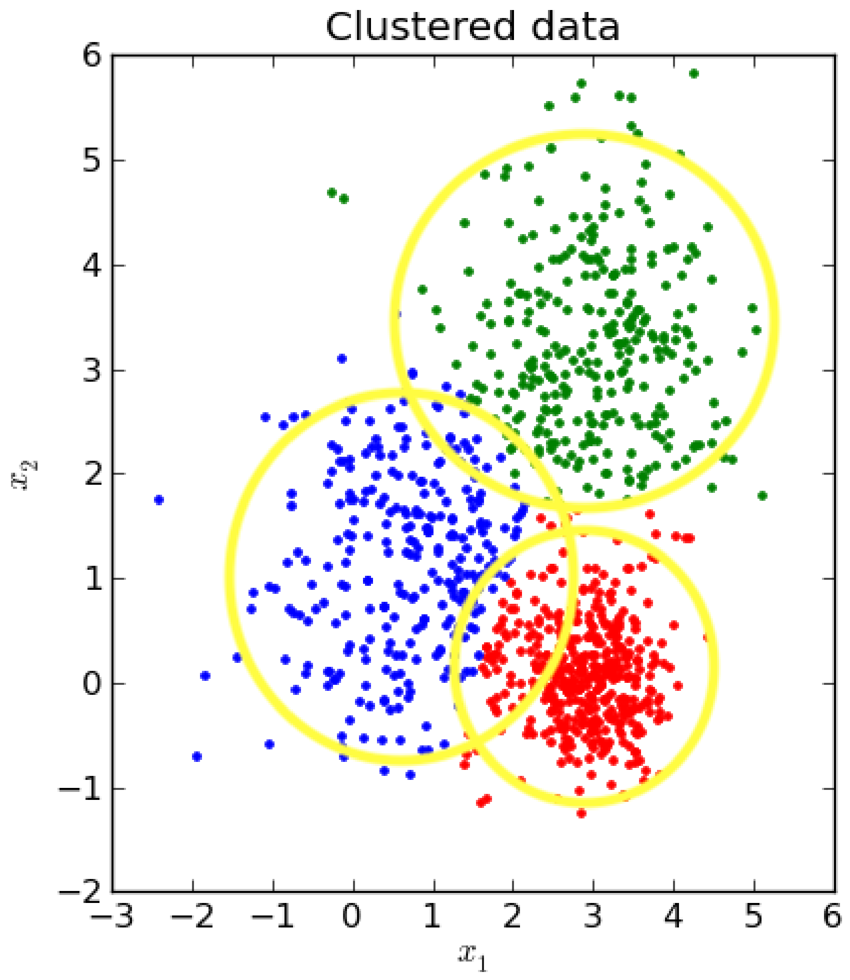
Starting with clustered data



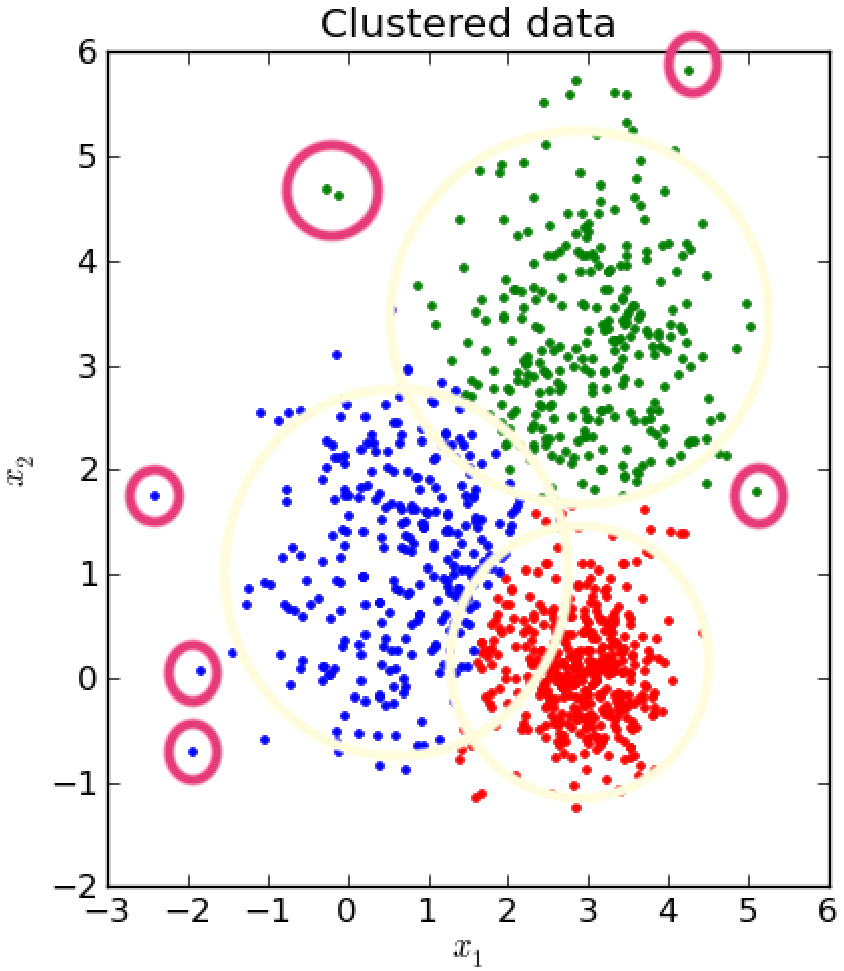
Assign the cluster centroids



Define distances from the cluster centroid



Flag fraud for those furthest away from cluster centroid



Flagging fraud based on distance to centroid

```
# Run the kmeans model on scaled data
kmeans = KMeans(n_clusters=6, random_state=42,n_jobs=-1).fit(X_scaled)

# Get the cluster number for each datapoint
X_clusters = kmeans.predict(X_scaled)

# Save the cluster centroids
X_clusters_centers = kmeans.cluster_centers_

# Calculate the distance to the cluster centroid for each point
dist = [np.linalg.norm(x-y) for x,y in zip(X_scaled,
X_clusters_centers[X_clusters])]

# Create predictions based on distance
km_y_pred = np.array(dist)
km_y_pred[dist>=np.percentile(dist, 93)] = 1
km_y_pred[dist<np.percentile(dist, 93)] = 0
```

Validating your model results

- Check with the fraud analyst
- Investigate and describe cases that are flagged in more detail
- Compare to past known cases of fraud

FRAUD DETECTION IN PYTHON

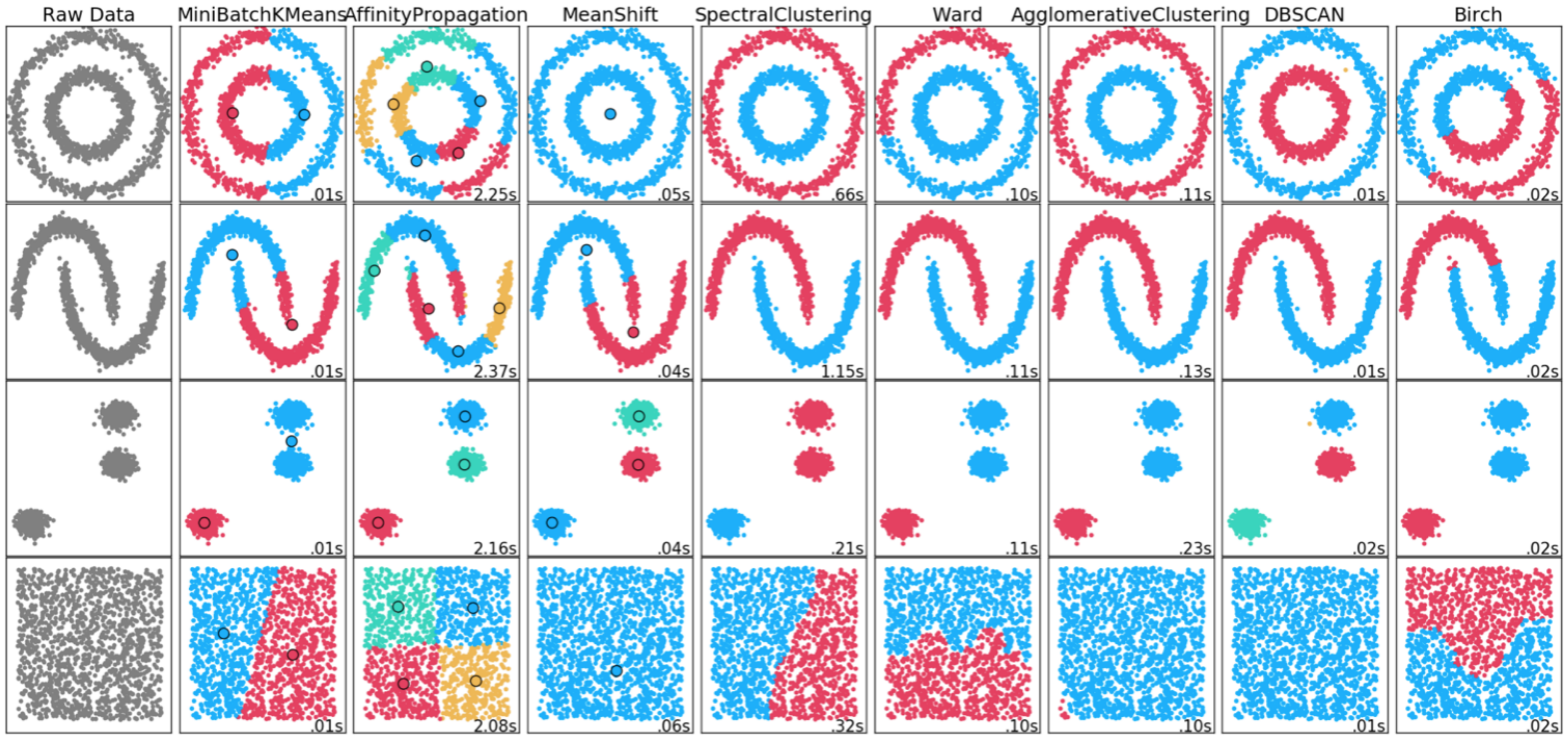
Let's practice!

FRAUD DETECTION IN PYTHON

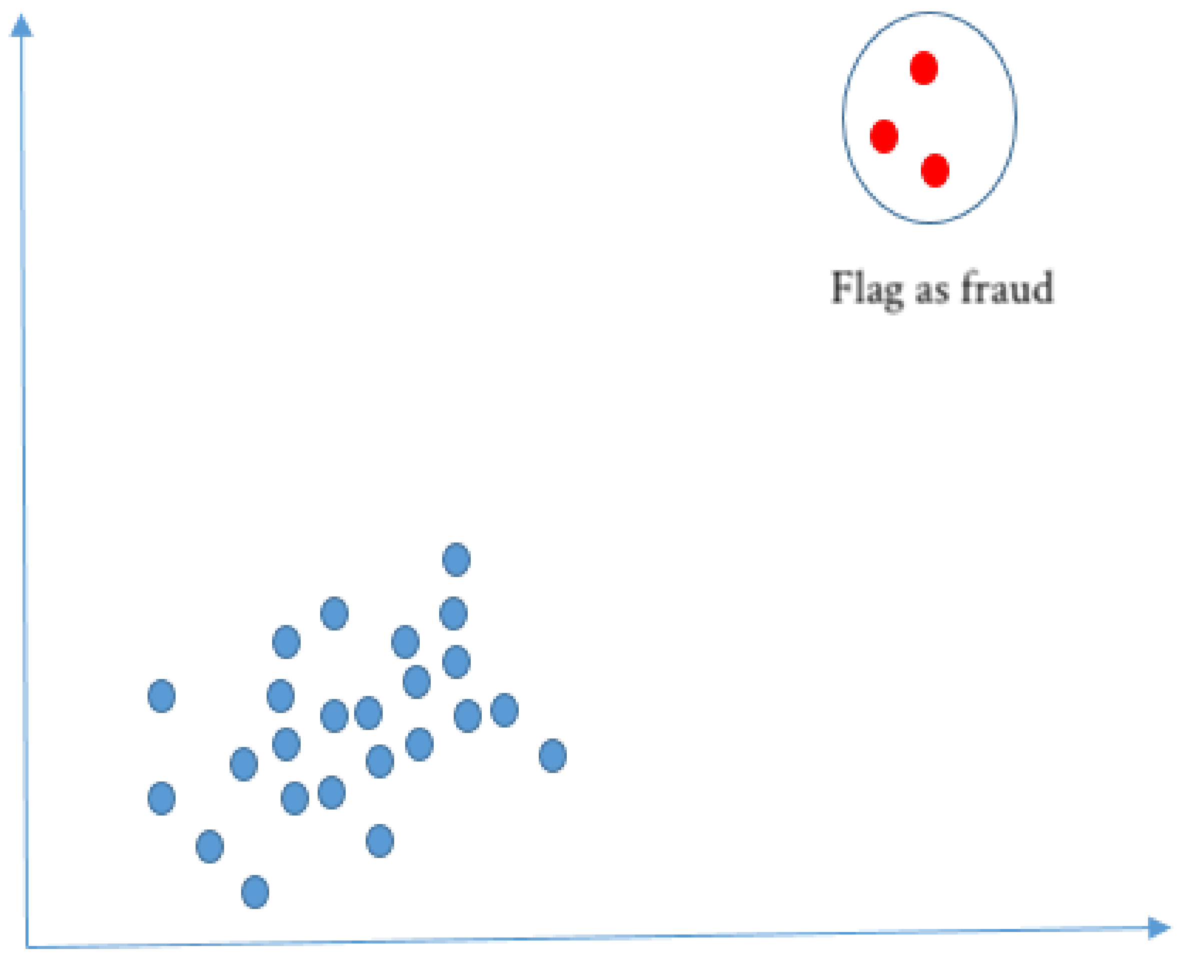
Other clustering fraud detection methods

Charlotte Werger
Data Scientist

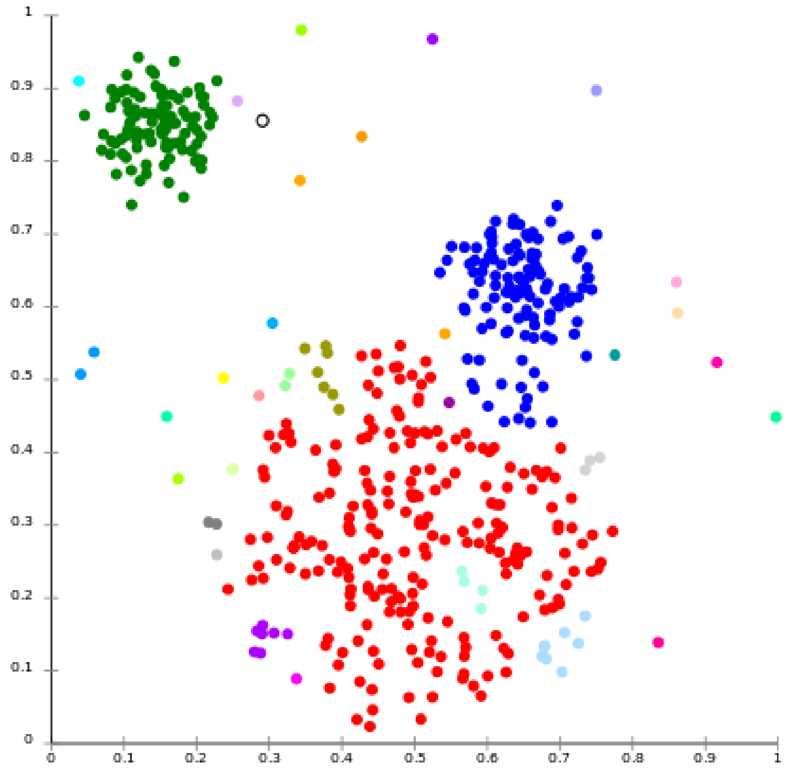
There are many different clustering methods



And different ways of flagging fraud: using smallest clusters



In reality it looks more like this



DBScan versus K-means

- No need to predefine amount of clusters
- Adjust maximum distance between points within clusters
- Assign minimum amount of samples in clusters
- Better performance on weirdly shaped data
- But..higher computational costs

Implementing DBscan

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.5, min_samples=10, n_jobs=-1).fit(X_scaled)

# Get the cluster labels (aka numbers)
pred_labels = db.labels_

# Count the total number of clusters
n_clusters_ = len(set(pred_labels)) - (1 if -1 in pred_labels else 0)

# Print model results
print('Estimated number of clusters: %d' % n_clusters_)

Estimated number of clusters: 31
```

Checking the size of the clusters

```
# Print model results
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X_scaled, pred_labels))

Silhouette Coefficient: 0.359

# Get sample counts in each cluster
counts = np.bincount(pred_labels[pred_labels>=0])
print (counts)

[ 763  496  840  355 1086  676   63  306  560  134   28   18  262  128  332
   22   22   13   31   38   36   28   14   12   30   10   11   10   21   10
    5]
```

FRAUD DETECTION IN PYTHON

Let's practice!